To everyone’s amusement, we spent much of the lecture time locked in combat with the AV equipment.

We did manage to trace through the execution of Dijkstra’s algorithm on a small example.

We briefly discussed the computational complexity of the algorithm. What follows here is a more in-depth analysis:

Assume the graph has $n$ vertices and $m$ edges.

All initializations take $O(n)$ time. The main while loop executes $\leq n$ times. On each iteration, we

- select the vertex in $C$ with the least $B()$ value
- examine all neighbours of the selected vertex – potentially performing some update work on each one

If $C$ is stored in an array, choosing the vertex with the least $B()$ value takes $O(n)$ time. Alternatively, if $C$ is stored in a min-heap, choosing the vertex with the least $B(v)$ value and repairing the heap takes $O(\log n)$ time.

Examining all neighbours of the selected vertex looks like it will take $O(n)$ time, but we can step back and observe that the total number of times we will examine a neighbour is $O(m)$ – since each edge gets considered at most twice.

If $C$ is stored in an array, the update work for each neighbour takes $O(1)$ time. Alternatively, if $C$ is stored in a min-heap, the update work for each neighbour takes $O(\log n)$ time since the neighbour may move up in the heap.
Putting it all together, we find that the two options for storing $C$ produce these results:

C in an array: 
- Choosing the next vertex: $O(n^2)$
- Examining and updating neighbours: $O(m)$

C in a min-heap: 
- Choosing the next vertex: $O(n \cdot \log n)$
- Examining and updating neighbours: $O(m \cdot \log n)$

Since $m \leq n^2/2$, the array implementation gives $O(n^2)$ time – this was Dijkstra’s original implementation.

If $m \leq kn$ for some constant $k$, the min-heap implementation gives $O(n \cdot \log n)$ time. Real world networks often have this property. However if $m$ is close to its upper bound, this min-heap implementation is less efficient than the array implementation.

The fastest-known implementation of Dijkstra’s algorithm stores $C$ in a structure called a Fibonacci heap (you may want to look this up). This implementation runs in $O(m + n \cdot \log n)$ for all graphs.